Deloitte.

Architecting the Cloud, part of the On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title: **Testing: as automation ascends, human play a critical role**

Description: As DevOps matures, and the "shift-everything-left" philosophy gains ascendancy, there's a movement to automate all—or most—phases of testing. However, there are some critical functions that may resist automation. In fact, contrary to the "automate-everything" impetus, human testers won't ever go away. Indeed, human testers need to be more involved, and earlier in the development process. In this podcast, Mike Kavis and guest, Angie Jones, discuss the human aspect of testing and how humans add significant value by assessing the system as a whole, helping developers design better code, and determining the level of testing automation that should occur. Angie also shares the testing automation matrix she has developed. Finally, they cover testing of machine learning algorithms—ways to help prevent or reduce bias and make algorithms more effective, and the emerging field of visual testing, which uses humans to ensure that graphics-heavy apps function and appear as designed.

Duration: **00:26:32**

Operator:
The views thoughts and opinions expressed by speakers or guests on this podcast belong solely to them, and do not necessarily reflect those of the hosts, the moderators, or Deloitte. Welcome to Architecting the Cloud, part of the On Cloud Podcast, where we get real about Cloud Technology what works, what doesn't and why. Now here is your host Mike Kavis.

Mike Kavis:

Hey, everyone. Welcome back to Architecting the Cloud podcast, where we get real about cloud technology. I'm Mike Kavis, your host and chief cloud architect over at Deloitte, and today I'm joined with Angie Jones. Angie's a senior developer at Applitools who specializes in test automation strategies and techniques. She shares a wealth of knowledge by speaking and teaching at software conferences all over the world, which we'll talk about in a little bit and as well as leading an online learning platform, Test Automation University. She also has about 25 patents across the world, U.S. and China, and in her spare time, and I'll put that in quotes, "spare time," Angie volunteers with Black Girls Code to teach young people coding workshops and attract more people to tech. So, Angie, welcome to the show. Tell us a little bit about your background and tell us about the good work you're doing at – with the coding, with the kids. And also tell us about your testing platform

Angie Jones:
Yeah, thanks for having me. So, I'm a test-automation engineer by trade. I've been doing this for quite a long time now, had a couple of sprints in product development, but most of my career has been in test automation. I recently transitioned to doing developer advocacy work, which is essentially still doing test automation, but I help other people everywhere around the globe to do their test automation better. So, that's via workshops or conference talks or blog posts and articles. So, it's a pretty cool job.

Mike Kavis:
Yeah, not bad. Tell us a little bit about the Black Girls Code initiative.

Angie Jones:
Yeah, so it's a wonderful organization that targets black girls between the age of 7 and 17, and the objective there is just to get more girls in the pipeline for tech jobs and expose them to technology. So, me, growing up, I didn't know what computer science was. I didn't know it was a thing. I didn't know of anybody who did this professionally, so I often think about that. I kind of stumbled into tech, just going to college, and my father recommended that I take some computer course, because he realized that this was an emergent space. And I took it, and I loved it, but had we not had that conversation, I might not have ever been exposed to this.

So, that's something that stuck with me as I moved into the workforce, and I wanted to make sure I was giving back and helping other girls like me realize that this is a viable option for a career. So, I volunteer with Black Girls Code. I used to lead a chapter in North Carolina, but since then I've moved, and now I just kind of volunteer with my local chapter, but there's chapters all over the U.S. There's also a couple in other countries as well, but we meet with the girls maybe once a month or so in the various different chapters, and we'll host a tech workshop and teach them hands-on things with working with tech. So, some of the workshops we've done, JavaScript or AI, block chain, robots, Raspberry Pi, all kinds of things. So, we expose them to all of these different things, game development, mobile development, and they really, really enjoy it. And it's in a nice, safe environment of their peers, and we think that really helps.

Mike Kavis:
Cool. We appreciate people give back to the community. Thanks for that, and thanks to your dad for getting you into tech. I'll tell you a funny story of how I got into computers. So, I'm aging myself a little bit, but when I was in high school, we got our first computer. So, this was the early '80s, no hard drive, all that stuff, and it was in the detention room, and I got in trouble, and I got sent to detention for a couple hours a day, pretty bored. And I'm like, "Hey, there's this computer over there." I started messing with it, and that's how I got into computers. So, I guess it's a good thing I got in trouble one day. I've been in trouble ever since. So, let's talk about testing. So, I'm glad you're on the show, because you're actually the first person in testing or specializing in testing I've had on the show, and I've been asking a lot of people. I've been struggling with, what's the future org structure of testing, because we're shifting all this testing left, and I think there's this misperception that we just write a bunch of test harnesses; we don't need testers anymore. That's far from the truth. I think that's more unit testing. There is a lot of testing that happens after that. So, what are you seeing the evolution of the operating model for – what's the organization look like in the future in this whole DevOps world, where we're releasing multiple times a day or a week, and we're shifting a lot of stuff left? What does that testing organization look like as we move forward?

Angie Jones:
Yeah, people have been trying to figure this out for quite a while, especially as we transition to different methodologies within development. So, as we went from waterfall to more shops doing Agile and now more shops doing DevOps, there's always that question of, is there still a need for a tester? And there was a period where quite a few companies were actually getting rid of their testers, especially the manual testers and trying to shift that work left. There's the buzzword of shift left, and everyone's talking about this within the concept of digital transformation. And what they're thinking that means is, "Yeah, our developers can do our testing." That's not quite working out so well. So, I'm sure you know this. Most developers don't really enjoy the act of testing, and so therefore they don't really do a thorough job, and I'm finding more than anything a lot of them don't really know how to test, right? That's not something we were taught when we studied computer science, or went to a boot camp, or whatever. This is not a topic that really came up much, and so the skill is not there. That's kind of lacking. But when we ask our developers to be the sole person responsible for testing their stuff, we notice that the quality drops, and this is across the industry usually. It's just a different mindset.

I worked in both development—I've worked in testing, and any time I'm doing development even now, I do a couple of side projects, where I'm doing development, I suck as a tester when I've developed those things. My mind is just not in that space at that time. So, what we're seeing now, and I think more towards the future, is, yes, the act of testing shifts left, meaning the tester is more involved. So, we still keep the testers, but we have them involved early on in the process. So, invite them to all of the meetings, the planning, the design meetings, when we're looking at all of the features and trying to determine how they will be executed or whatever. We invite them, and the testers can actually test that early in the phase. They don't have to wait until the code is done. I've seen brilliant testers be able to listen to the design of something and be able to poke holes in it right then and there and find bugs right there before any code is ever written. And this is the act of shifting left. So, it just means testing earlier, testing before the code is done. But, please keep your testers. You definitely need them.

Mike Kavis:
Yeah, and I think there's a couple factors here. One is as we move to microservices and these more modern technologies, the developers focus on a service, but they don't necessarily have a vision of the entire platform or the entire application or service they're building. They're just focused on this service, so they're writing a test harness that says, "I'm getting these inputs, and I expect these outputs," and that's the extent of it, and that's the extent of the

automation of the build process. So, they're ensuring that their service works as expected, but they're not ensuring that the system works as expected. And I think that's where the testing organization can add a lot of value.

Angie Jones:
Yeah, exactly. So, I love for developers to write unit tests for whatever module or feature or service that they are working on, but to ask them then to think about tests you can write or end-to-end integration tests that kind of make sure that this works with other components or other services is a bit far-reaching. It sounds good in theory, but that's not going to work out as well. Whereas testers, they kind of have this overall view. That's one of the reasons why, even though I did the development sprints, I much preferred to do my coding in the testing space. I love coding, but I also have this tester's mindset, where I want to see the big picture and how all of the different pieces fit together. I think I can empathize really nicely with customers and users, so that helps me come up with different scenarios of how they might use it. So, all of that is something that you need your testers for.

Mike Kavis:
Shift left is applying to a lot of things. We're shifting security left. We're shifting ops left. We're shifting test left. But what that's also doing is it's forcing us to design for test, design for ops, design for security, where traditionally these were handoffs. And then when it got to these other departments, they said, "Well, what about our requirement?" right? And I think as you shift people earlier in the lifecycle, you're getting those requirements up early. So, people are designing for the endgame. Instead of just passing errors down the path, and it's more expensive to fix them at the end, you're actually addressing that in design, and I think that's the whole mantra of shift left to me isn't that people are going away; it's that you're including all these different domain experts at the beginning, and you're designing a better product. Are you seeing that?

Angie Jones:
Yes, yes, very much so, especially in the Agile, DevOps type of spaces or shops. We have to automate test, so doing the manual testing is not enough. If we want to take advantage of continuous integration, or we want to deploy daily, multiple times a day, then those tests have to be automated. And in that, that means that the application itself needs to be automatable and needs to be test-automatable, meaning developers have to think about these things. So, that's why it's also so important to have your testers and your test-automation engineers present very early. I sit with my devs during design phases and say things like, "Okay, what is it that you're building? Let's kind of mock this out. And from there, let me tell you all of the different code things I'm going to need, I need you to develop these things in, because I need to automate against that." So, maybe I need an API to be able to ensure that the setup is done quickly and I'm not doing that via the user – the GUI, for example, right? Or if we're going to use the GUI for the test, then I need these identifiers on your elements and things like this. So, these are things that they have to consider not just – like you said, not just for testing but for ops and everything else as well.

Mike Kavis:
Yeah, I agree. I'll give you an example. Back in the day when I was doing this stuff, we were sending information to grocery stores. We were doing coupons at point of sale, but everything was compressed and binary, because you had to minimize what you're sending. Well, you can't test looking at binary, so we had to write a bunch of tools for the testers, right, to convert that to ASCII so you could test. So, I think that's what you're referring to. When you're in the design phase, you're like, "Okay, I need to test this. Therefore, build me these tools, or create me this API, or give me the ability to actually validate this."

Angie Jones:
Right, and it's important that everyone hears that. This shouldn't be a side conversation with a tester and developer, because the business needs to know that, okay, in order to develop this feature, I also have to make it testable, right? And that might require a little bit more than simply hacking out the feature itself. And so, making sure that everybody understands this and is on board with this really helps.

Mike Kavis:
Yeah, and you mentioned automation, and I watched a presentation you did when you were at Twitter about automate all the things and the perceptions that you have to automate all of the things. And you put together a pretty good framework for figuring out what you should and shouldn't automate, so talk a little bit about that.

Angie Jones:
Yeah. So, I talk mostly to practitioners, so people like me who are actually in the trenches doing the work, right? And most of those people understand that it's not that great an idea to automate all of my tests. But leadership doesn't always feel the same way. So, when I talk to leaders and when I've worked as an IC in places, we see that the managers are like, "Okay, all I know is I want to have several deployments this week. That means the tests need to be automated. We don't have time for any exploratory testing or running these same tests over and over again by a human being. We want it all automated." And that's the viewpoint they're coming from. But there's a lot of drawbacks to automating all of your tests. It's a lot of noise. It becomes slower, so if you're using this for continuous integration or CD, then you want your test to be as fast as possible. Even running these in parallel, you run into challenges when it becomes thousands of these things. And so, it's just not that great an idea, and so I came up with a matrix on how you decide what to automate, and I used this at Twitter.

The example that I do in that talk is on some of the tests that I had to evaluate for Twitter, and it basically uses four different components, and with that you give it a score, a grade, essentially, for each test. So, the four different components are risk, value, cost-efficiency and then historical data on that feature area, like, "How often is this area breaking? How many defects do we have around this area?" kind of thing. And then you put all of that together, and you come up with a score. So, with this, now if you grade all of your tests, you have basically a score on all of them. You can sort these, and you can see, okay, this bottom 25 percent, let's not even bother trying to automate these. It doesn't seem like it's worth it, and the top 25 percent is your go-to. This is where you shift focus on. And the great thing about it is because it's a numerical value, you can sort these, and now you know, okay, this is the most important thing to automate right now, and you just go down this list. So, a couple of shops are using it, and they found a lot of value in it.

Mike Kavis:
Yeah, I saw the presentation. It was pretty cool. I mean, I think it's important to tell the listeners that most of the tests will get automated, just not all of them, but your first pass through, you identified a couple that you didn't think had value and one that you checked as, "Yeah, we should do this." Then when you went through your matrix, it wound up that one of those just – it's never been broken and at really low value, so that one fell out. And one of the

ones you thought, “Nah, we don’t want to do that,” wound up being really critical. So, I think it’s so much better sticking your finger in there, saying, “I think it’s this one.”

Angie Jones:
Right, because as humans we look at these things, and we kind of just assume all of them need to be automated. Even people who say we shouldn’t automate all the things, if I give them a list and I say, “Here are your scenarios. Which one should we drop?” they never want to drop any of them, because it feels like anything that we’re testing feels like it’s important. But when you look at the different criteria, you start to realize now you have something tangible. And you can take this to leadership. They love charts and graphs and numbers and stuff. You put something –

Mike Kavis:
Make it blue, yellow, and green, and they’re all in.

Angie Jones:
Yes, slap some color on that thing, exactly, and then you get your buy-in.

Mike Kavis:
I’m not even going to really ask a question here other than, how the heck do you test machine learning?

Angie Jones:
So, a lot of people think that you don’t test machine learning is what I see. So, when I talk to people – and I’ve worked places where we’ve done machine-learning features. We’ve done some at Twitter. We’ve done some at another company I worked at, and any time there’s a feature that’s utilizing machine learning, what you hear is, “We don’t have to worry about testing this feature. That’s the machine-learning feature,” and everybody just kind of goes, “Oh, okay,” and just like that makes sense. And we’re treating this like this magical black box that just works, but we know that it doesn’t always work. We see news reports all the time of goof-ups.

So, you have to test these things, and people usually think you don’t have to test it, and then when I ask, “Okay, well, we do want to test it. How do we test it?” It’s kind of like, “I don’t know.” So, when I had to do the testing of it, I just kind of, one, make sure I understood how the system is learning. So, I give examples to people, and I ask things like, “How would you test your Netflix recommendation page?” That’s driven by machine learning. How do you know that the stuff there is the stuff that should be shown to you or recommended to you?

And no one can answer, “I don’t know. I don’t know how I would test it,” right? That’s because we don’t really understand what’s going on behind the scenes and what rules they have in place. So, once you understand what’s the rules for recommending to someone, that helps a lot. That’s Step One. Step Two is then training the system on what it is it’s supposed to learn, so coming up with some scenario. And it’s really critical to have a diverse set of data to feed to the system so that it’s learning what it is you wanted to learn and not something else goofy. And then, finally, once you’ve trained the system, you have another set of data that you can then test it with to make sure that it is making the right predictions or recommendations.

Mike Kavis:
Yeah, and I think the need here is for domain expertise to validate the reasonableness of the answer. So, I’ll give you an example, let’s say in health care, and it does an analysis, and it says you should recommend this to the patient. Well, as developers, we’re like, “Sounds good to me. I don’t know.” But put a doctor or a nurse in there, and they look at that and go, “No, that’s just wrong.” So, the machines are learning based on what we’re feeding it. I think the domain experts help us train the machines the right way.

Angie Jones:
Yeah, that is such a great point, because one of the things that I found so different in testing machine learning than other things is that there’s no exact answer here. There’s no correct – just exact correct answer, especially when I was trying to automate against this, because I have to write assertions. So, what do I assert against? What am I expecting here? And what I learned is, there’s just a range of validity, and knowing what that range is really helps. And, like you said, domain expertise I think is critical, because common sense needs to trump these algorithms, right? You need to know, okay, this gave me this back, but does that make sense? And execute your judgment there, and it would really help if you had someone who was an expert in that domain, where this stuff is just common sense to them, where it might not be for us.

Mike Kavis:
Yeah. I’ll give you a good example. Several years ago, when there was a race for who’s going to win the music app, right. And I’m a metal-head, right, and Iron Maiden, Judas Priest guy, and they recommend, “Go listen to Huey Lewis and the News.” I’m like, what? And then my recommendations were always off, and I actually left that platform, because I like to – just tell me what I want to listen to. Introduce me to new stuff, and they’re giving me pop and stuff I don’t like, people really need to think about this. If you’re depending on machines, you need to validate that the machines know what they’re talking about.

Angie Jones:
Exactly.

Mike Kavis:
We had had a discussion the other day about functional testing versus visual testing. Explain a little bit what you mean about visual testing and why that’s so important.

Angie Jones:
Yeah. So, it’s a relatively new concept in test automation. It’s essentially a superset to functional testing. I’ll give you an example. So, I was on Instagram the other day, and I saw this sponsored post. So, Instagram has ads. You can buy a post, and it’s embedded in people’s timeline, or whatever. And of course you pay for this sponsored post. So, I was looking on Instagram. In this sponsored post, there was no picture there. Instagram is all pictures. There was no picture there. All the text was bunched up, overlapping each other in the upper-left corner, all right? So, I got to thinking about this. As a human being, we

can easily see this, but like we talked about a little bit earlier, most of our tests are being automated now. So, you're not going to have someone check that manually every time you want to deploy something. So, that's going to be automated, and that automated test is probably checking to see, is this text present? True. Is this text present? And all of the text that's present, everything is true, so all of the assertions would pass, and that functional test would pass, even though visually this is very, very wrong and a very expensive bug, right?

So, the visual assertions – my company, Applitools, does visual assertions. They have an API, and what this does is takes a picture of the app at the given point when you write the test. If everything looks good at that point, then that's fine. It saves that as a baseline, and then every time this test runs as part of regression, then it takes another picture and does the comparison. And it doesn't do pixel-to-pixel comparison, because that's really flaky. Instead, it uses machine learning to be able to knock out the things that we don't care about as humans like little white-space shifts or something like that. It doesn't catch those things. It only catches the things that we care about as a tester or developer. So, it's really interesting, and with that, and with taking a screen shot, you're essentially taking – you're doing all those functional assertions, too, right? So, you're making sure the text is present and all of that, so you kind of could get rid of that, and you just do this one screen-shot, and it's like, "Yep, I validated all the things you care about and more, really."

Mike Kavis:
And I think that drives home our earlier point, where the test harness is doing a unit test that's automated in the CICD process can only cover so much. You still need to do that full integration testing. You still need to look at all the different devices and endpoints, and that stuff's never going to go away.

Angie Jones:
Right, right, and that's where you see a lot of these goofy visual bugs is on the different devices, because the viewport changes, right, and that's where people kind of mess up. I see a lot of applications get caught up right there, and so that's why the visual piece is important.

Mike Kavis:
Great. Well, really enjoyed the talk today. I think I need to get more testers on this show. That was fun.

Angie Jones:
Thank you for having me.

Mike Kavis:
Yeah, so I guess we can catch you on Twitter @techgirl1908

Angie Jones:
Yes, that's me.

Mike Kavis:
– your website's AngieJones.tech. What're we going to find there?

Angie Jones:
I write about test-automation strategies and techniques, so it's really interesting there. I also run Test Automation University, so if any developers are looking to learn more about test automation or if you have testers who want to learn, or if you have people who're also in test automation, but they want to just kind of up their skills, this is a free platform with all courses on test automation taught by the leading experts in the industry. So, it's community-driven. Everything is free. It's a really valuable resource.

Mike Kavis:
And you just hit a major milestone on that, I think I saw on Twitter. What was that milestone?

Angie Jones:
Yeah. So, we launched this in January of this year, so within 6 months we hit 20,000 registered users, so pretty excited about that.

Mike Kavis:
Cool. You might have 20,001 now. I have to go check that out. So, thanks again for stopping by and giving us your insights. To learn more about Deloitte or read today's show notes, head out over to www.DeloitteCloudPodcast.com, where you'll find more podcasts by me and my colleague Dave Linthicum just by searching for Deloitte on Cloud podcast on iTunes or wherever you get your podcasts. I'm your host, Mike Kavis. You can find me @madgreek65 at Twitter, or you could e-mail me directly, MKavis@deloitte.com. Thanks for listening, and we'll see you next time on Architecting the Cloud.

Operator:
Thank you for listening to Architecting the Cloud with Mike Kavis. Connect with Mike on Twitter and LinkedIn and visit the Deloitte On Cloud blog at www.deloitte.com/us/deloitte-on-cloud-blog. Be sure to rate and review the show on your favorite podcast app.